

REMARKS

In response to the Office Action mailed on May 4, 2006, Applicants respectfully request reconsideration. In furtherance of prosecuting this patent application, Applicants submit the following remarks discussing patentability of rejected claims 1-35 as set out in the present Office Action and include new claims 36-43.

Objections to Claims 2-10, 12-14, 16-24, 26-28, and 31-35

In accordance with the Examiner's suggestion, Applicants have amended the "objected to" claims.

Rejection of Pending Claims 1-35

The Examiner rejects claim 1 under 35 U.S.C. §103(a) as being unpatentable over Young (U.S. Patent 6,782,531) in view of Million (U.S. Patent 6,742,176). The Office Action likens elements in Young and Million to those in claim 1 to reject the claimed invention.

Applicants have reviewed the pending claims in view of the cited references and respectfully submit that the claimed invention includes distinguishing limitations over the cited art.

For example, the Examiner submits that figure 4 and corresponding text at column 9 lines 12-16 discloses the step of "retrieving a dependency list indicating respective plug-in services provided by, and required by, each plug-in module identified in the identities of a plurality of plug-in modules." However, Applicants point out that pending claim 1 includes a limitation not addressed by the Examiner. For example, claim 1 recites retrieval of the dependency list "based on queries to the plug-in modules." Claim 1 was amended on June 1, 2005 to include this limitation.

Accordingly, the claimed technique recites retrieving a dependency list based on queries sent to the plurality of plug-in modules. The passage cited in Young merely indicates that the execution management framework 425 in Young

utilizes respective dependency information from configuration file 418 as indicated at column 9 lines 5-33:

FIG. 4 shows an architectural view of a pipeline stage 400. The stage 400 includes an input queue 402 (e.g., a FIFO buffer), a multithreading process space 404, and an output queue 408 (e.g., a FIFO buffer). The process space 404 processes a number of plug-ins, numbered 1 through 8, under the control of an execution management framework 412, and pursuant to processing threads stored in a thread pool 414. A stage configuration module 416 receives configuration files 418 from the configuration manager 150, which define stage operations as well as operation of the plug-ins nos. 1-8 of the corresponding process space 404 and their processing interdependencies. The stage configuration module passes the plug-in configuration information to an execution management framework 425. The execution management framework 425 uses this information to determine which of the plug-ins nos. 1-8 can be processed in parallel (and during the same clock cycles per clock 420) and which of the plug-ins nos. 1-8 need to be processed in sequence after other plug-ins because they depend on a final or intermediary result from the other plug-ins. As illustrated, those of the plug-ins nos. 1-8 that can be processed in parallel are shown in a vertical stack in the drawing (as in the case, e.g., of plug-ins 1, 2 and 3; or plug-ins 6 and 7; or plug-ins 8 and 9). Moreover, those of the plug-ins nos. 1-8 that are dependent on, and therefore need to be processed after, other plug-ins are shown to the right of the other plug-ins on which they depend (as in the case, e.g., of plug-in 5 dependent on plug-in 4 and thus shown in the drawing, to the right of plug-in 4). (Emphasis added)

In view of this cited passage, claim 1 recites a technique that operates in an opposite manner as the cited subject matter used to reject the claimed invention. That is, claim 1 does not recite retrieval of dependency information from a separately maintained configuration file. Instead, claim 1 indicates that the queries are sent to the plug-in modules to retrieve dependency information.

-18-

Applicants submit that figure 4 illustrates that the plug-in modules 1-9 reside at a remote location with respect to the configuration file 418. Accordingly, the execution management framework 425 in Young does not perform a query to the plug-in modules to identify a respective ordering.

This claimed technique enables the plug-in modules themselves to provide dependency information of what other plug-in modules to execute rather than relying on efforts of the execution management framework 425 as discussed in Young to track such information for all plug-ins via a respective configuration file 418. Thus, Applicants respectfully request the withdrawal of the respective rejection of claim 1 under 103(a).

For the reasons stated above, Applicants submit that claim 1 includes limitations not found in any of the cited references and therefore is patentably distinct and advantageous over the cited prior art. Applicants respectfully request the withdrawal of the rejection of claim 1 under 35 U.S.C. §103(a) or request that the Examiner more particularly point out passages in Young that teach a technique of obtaining a respective dependency list based on queries to the plug-in modules. Accordingly, Applicants respectfully request allowance of claim 1.

Because claims 2-14 depend from and further limit claim 1, Applicants submit that claims 2-14 are in allowable condition as well.

Claim 15 includes similar limitations as discussed above for claim 1. Accordingly, Applicants respectfully submit that claim 15 includes similar patentable distinctions over the cited prior art as does claim 1. Applicants respectfully request allowance of claim 15 as well as corresponding dependent claims 16-28.

In a similar vein as claim 1, Applicants would like to point out that claim 29 recites "querying a dependency interface associated with the plug-in module with

a dependency query to obtain a dependency response from the plug-in module, the dependency response indicating respective plug-in services provided by the plug-in module.” As discussed above, the cited passages in Young do not disclose querying a plug-in module to learn of plug-in services provided by the plug-in module as discussed above. Accordingly, Applicants respectfully request allowance of claim 29. By virtue of dependency with respect to claim 29, claims 31-35 also should be in condition for allowance.

Applicants respectfully submit that the pending dependent claims include further patentable distinctions over the cited art.

For example, claim 3 recites “receiving a dependency response from the plug-in module...” The Office Action cites column 9, lines 12-16 to reject this portion of claim 3 of the subject application. The cited language in Young reads as follows:

A stage configuration module 416 receives configuration files 418 from the configuration manager 150, which define stage operations as well as operation of the plug-ins nos. 1-8 of the corresponding process space 404 and their processing interdependencies. (emphasis added)

Applicants respectfully submit that this passage provides no literal support or suggestion that any plug-in module as in Young provides “a dependency response from a plug-in module...indicating respective plug-in services provided by, and required by, the plug-in module” as in the claimed invention. That is, the “configuration file” in Young resides at a separate location with respect to the plug-in modules. Specifically, corresponding text associated with FIG. 6b of Young indicates that the dependency data 654 resides in the execution management framework and not in a respective plug-in module. There is also no

indication in the cited Young reference that any plug-in module itself provides an indication of services supported by the plug-in module.

In a similar vein as discussed for claim 1, the technique in claim 3 enables the plug-in modules themselves to provide dependency information of what other plug-in modules to execute rather than relying on efforts of the execution management framework to track such information for all plug-ins via maintaining a configuration file. Thus, Applicants respectfully request the withdrawal of the respective rejection of claim 3. Claim 17 includes similar limitations as claim 3 and should be allowable for similar reasons.

Applicants submit that claim 5 includes limitations not disclosed by Asazu. For example, claim 5 recites "querying a dependency interface ... to obtain the dependency response from the plug-in module." As recited in claim 3, the dependency response indicates respective plug-in services provided by and required by the respective plug-in module.

The Office Action cites column 11, lines 60-65 to reject this claim limitation:

In the preferred embodiment of the present invention, these services is separated from the package body as the QT plug-in component 14, and the query interface package 12 itself is configured to provide only the management and/or look-up functions of the QT plug-in component 14, permitting dynamic addition of various services in future.

This cited passage merely indicates that a plug-in module includes a query interface for enabling management of look-up of functions of a respective plug-in component. This presumably enables a remote entity using the plug-in module to call functions in the plug-in module. The passage does contemplate that the query interface can be used to add services in the future. However, contrary to the Examiner's assessment, Applicants respectfully submit that this

provides no indication that the query interface of a respective plug-in module is used for purposes of providing a dependency response as in the claimed invention.

Where does Asazu specifically recite that a respective query interface of a plug-in module is used for purposes of providing dependency information?

Thus, Applicants respectfully request allowance of claim 5. Claim 19 includes similar limitations as claim 5 and should be allowable for similar reasons.

Claim 9 recites "forwarding, via a dependency available interface associated with a respective plug-in module, a list of initiated plug-in services of other plug-in modules that are currently available for use by the respective plug-in module." To reject this claim, the Examiner cites Young at column 8, lines 41-43 which states:

Plug-ins 220 preferably have a system services interface for accessing system services provided via the framework 225, e.g., to allow the plug-ins 220 to read configuration data.

Applicants traverse the rejection on grounds that the cited passage in Young recites a respective technique that teaches away from the claimed invention.

First, the cited passage recites that the plug-ins include an interface to access the framework 225 to read configuration data. The claimed invention recites that the plug-in module forwards a list of plug-in services. These flow of information as disclosed by Young is opposite of that in the claimed invention. For example, Young recites that information is retrieved by a respective plug-in module. The claimed invention recites information being forwarded from a respective plug-in module.

Additionally, note the type of information being communicated is different as well. For example, the claimed invention recites forwarding a list of initiated

plug-in services of other plug-in modules that are currently available for use by the respective plug-in module. The cited passage in Young indicates that a respective plug-in module includes a service interface to read configuration information. There is no teaching or suggestion that the configuration information identifies a list of initiated plug-in services of other plug-in modules that are currently available for use. Accordingly, Applicants respectfully submit that claim 9 is also patentable over the cited art. Claim 23 includes similar limitations as claim 5 and should be allowable for similar reasons.

Claim 10 recites wherein “the step of initiating service operation of plug-in modules according to the plug-in initiation order comprises performing, for each respective plug-in module in the plug-in initiation order, the steps of:

determining, from a published list of services available by initiated plug-in modules, an identity of each initiated plug-in service required by the respective plug-in module;

forwarding to the respective plug-in module, via a dependency available interface associated with the respective plug-in module, the identity of each initiated plug-in service required by the respective plug-in module;

receiving a list of services initiated by the respective plug-in module; and

adding the list of services provided by the respective plug-in module to the published list of services.”

The Office Action cites column 12, lines 17-23 of Asazu to reject claim 10 of the subject application. The cited language in Asazu reads as follows:

FIG. 15 shows the QT plug-in descriptor 41. The QT plug-in descriptor 41 shown in FIG. 15 contains various information such as the kind of service specified as a Java class/interface provided from the QT plug-in component, the developer and version information, and the application program 8 is configured to retrieve the required QT plug-in component on the basis of these information.

The claimed invention includes many distinctions not disclosed or suggested by this cited passage. For example, this passage merely indicates that the application 8 is configured to retrieve a respective plug-in module based on description information.

However, the claimed invention recites a technique of forwarding information to the respective plug-in module. The information includes the identity of each initiated plug-in service required by the respective plug-in module. The cited passage in Asazu only indicates existence of respective of plug-in module information and that the application uses the information to retrieve plug-in module based on this information. This is not equivalent to notifying a respective plug-in module of which initiated plug-in module are required by the respective plug-in module.

Also, claim 10 recites receiving a list of services initiated by the respective plug-in module; and adding the list of services provided by the respective plug-in module to the published list of services. There is no indication in the cited passage that the application or any other resource receives information from a respective plug-in module, especially information such as services provided by the plug-in module for purposes of updating a respective information base.

This claimed technique enables the plug-in modules to manage themselves based on knowing which plug-in module from which they depend. The plug-in modules in Young only understand at what point in time they are to execute and not from which other plug-in modules they depend. Thus, Applicants respectfully request that Examiner allow claim 10 over Young. Claim 24 includes similar limitations as claim 10 and should be allowable for similar reasons.

Claim 13 recites "wherein the first plug-in module is initiated via the step of initiating operation of plug-in modules after initiation of the second plug-in module, and wherein the second plug-in module includes a wait-state operation



causing the second plug-in module to wait to provide the service offered by the second plug-in module until initiation of the first plug-in module.”

The Examiner cites Young at column 9 lines 24-33 to reject the claimed invention, which reads as follows:

As illustrated, those of the plug-ins nos. 1-8 that can be processed in parallel are shown in a vertical stack in the drawing (as in the case, e.g., of plug-ins 1, 2 and 3; or plug-ins 6 and 7; or plug-ins 8 and 9). Moreover, those of the plug-ins nos. 1-8 that are dependent on, and therefore need to be processed after, other plug-ins are shown to the right of the other plug-ins on which they depend (as in the case, e.g., of plug-in 5 dependent on plug-in 4 and thus shown in the drawing, to the right of plug-in 4). (Emphasis added)

Applicants submit that there is no indication that any of the plug-in modules shown in FIG. 4 of Young insert a wait state as in the claimed invention. This passage in Young indicates increasing time to the right. Vertically stacked plug-in modules in Young can be executed in parallel. Plug-in module 4 must wait for completion of plug-in modules 1, 2, and 3. Plug-in module 5 starts after completion of plug-in module 4, and so on. There are no plug-in modules that are executed in parallel that are dependent on each other. For example, dependent plug-in modules are executed after completion of a plug-in module on which they depend.

The claimed invention recites that dependent plug-in modules can be executed in parallel based on use of a wait state. That is, the first plug-in module is initiated after a second plug-in module and a wait state causes the second plug-in module to wait to provide the service until after initiation of the first plug-in module. Thus, the first plug-in module depends on a service provided by the second plug-in module and, even though a dependency exists, both plug-in modules at least partially execute in parallel.

Based on the recited technique in claim 13, there is no need to initiate execution of the plug-in modules in any particular order, even though a dependency exists. Young makes no mention of initiating execution of plug-ins in the order as described in the claimed invention. Nor does Young discuss a wait type of operation associated with a plug-in to receive services offered by another plug-in. Providing a wait-state operation in Young would serve no useful purpose because it would not be necessary.

Accordingly, Applicants respectfully request allowance of claim 13. Claim 27 includes similar limitations as claim 13 and should be allowable for similar reasons.

Claim 30 includes patentable limitations similar to the limitations of pending claim 13. For applicable reasons as discussed above for claim 13, Applicants submit that claim 30 is patentable over the cited prior art. Claims 31-35 depend from claim 30 and therefore also should be allowable over Young and the other cited references.

#### Claims 31-35

Applicants submit that dependent claims 31-35 include limitations not found in the cited prior art.

For example, claim 31 recites “querying a set of plug-in modules to identify services provided by the set of plug-in modules.” The Examiner cites column 11, lines 60-65 in Asazu to reject the claimed invention, which reads as follows:

In the preferred embodiment of the present invention, these services is separated from the package body as the QT plug-in component 14, and the query interface package 12 itself is configured to provide only the management and/or look-up functions of the QT plug-in component 14, permitting dynamic addition of various services in future.

Note that the query interface package 12 and plug-in component 14 are shown and discussed with respect to FIG. 3 of Asuza. The query interface

package 12 and plug-in module 14 are separate entities. The query interface package 12 is an entity providing management and look-up with respect to the plug-in module 14. There is no indication that the query interface package 12 communicates with the plug-in module to identify services provided by the plug-in module 14. Accordingly, Applicants respectfully submit that claim 31 includes limitations not taught or suggested by the cited prior art.

Claim 32 recites “in response to querying the set of plug-in modules, identifying plug-in modules not identified by the software application as being necessary but which are identified by the set of plug-in modules as being necessary to carry out execution of an operation on behalf of the software.” The Examiner cites column 11, lines 60-65 in Asazu to reject the claimed invention, which reads as follows:

In the preferred embodiment of the present invention, these services is separated from the package body as the QT plug-in component 14, and the query interface package 12 itself is configured to provide only the management and/or look-up functions of the QT plug-in component 14, permitting dynamic addition of various services in future.

The Examiner also cites column 12, lines 1-6 in Asazu to reject the claimed invention, which reads as follows:

The QT plug-in component 14 itself may be created as a component of GUI, permitting the functions required for the application program 8 to be constructed easily inclusively of GUI by only combining a plurality of QT plug-in components together as shown in FIG. 14 showing the QT plug-in components.

In short, contrary to the Examiner’s assessment of the cited passage, there is no indication whatsoever of querying multiple plug-in modules. Further,

there is no indication of carrying out a step of identifying plug-in modules not identified by the software application as being necessary but which are identified by the set of plug-in modules as being necessary to carry out execution of an operation on behalf of the software. In fact, as discussed above, the plug-in modules provide no such information in Asazu. Accordingly, Applicants respectfully submit that claim 32 includes limitations not taught or suggested by the cited prior art.

Claim 33 recites that “the third plug-in module initiating a wait state operation causing the third plug-in module to wait to provide the service offered by the third plug-in module until instantiation of the fourth plug-in module.” The Examiner cites column 13, lines 43-58 in Young to reject the claimed invention, which reads as follows:

The execution management frameworks 522, 524, 526 collectively constitute an infrastructure that allows any type or number of plug-ins to be arranged and operated in any order pursuant to a configuration file associated with each session. The infrastructure allows plug-ins to be distributed across multiple machines and takes care of the communication across processes and across machines. Some plug-ins have dependencies on certain properties being available at the time they operate on a session. They, in turn, can supply properties on which other plug-ins may depend. The infrastructure ensures that all required properties are available at the time the plug-in runs. Sometimes it does not matter which one of a group of plug-ins is called at a certain time because none of them have cross-dependencies on any of the others in the group, and then the plug-ins of the group can be run in any order.

Contrary to the Examiner’s assessment of the cited passage, Young does not teach or suggest that depending plug-in modules can be run in any order if they are dependent on results from another plug-in module. In fact, Young discloses just the opposite. Young does indicate that plug-in modules can be run

in any order as long as there are no dependencies. However, this is not what Applicants claim as their invention. The claimed invention recites use of a wait state to enable depending plug-in modules to execute at least partially in parallel. Accordingly, Applicant respectfully request allowance of claim 33.

Claim 34 recites “wherein the processor initiates execution of the first plug-in module before execution of the second plug-in module, the first plug-in module initiating a wait state operation resulting in signaling to the second plug-in module, the signaling indicating that a respective service of the first plug-in module is not yet available to the second plug-in module.” The Examiner cites column 14, lines 15-27 in Young to reject the claimed invention, which reads as follows:

A complexity is introduced in specifying that order because plug-ins can be dependent on other plug-ins. Generally speaking, given two plug-ins M and N, if plug-in M computes the value x (as a final or intermediary result) and plug-in N requires the value x to perform its computation, then plug-in N depends on plug-in M. Plug-ins can be dependent on zero, one, two, or more, other plug-ins. In the above notation, because of the noted dependency between M and N, the stage infrastructure will wait for plug-in M to be executed before it starts execution of plug-in N. Plug-ins with zero dependencies can be executed immediately or at any other time, without regard to prior execution of other plug-ins.

Contrary to the Examiner’s assessment of the cited passage, Young discloses that plug-in modules must complete execution before starting execution of another plug-in module when there is a corresponding dependency. For example, there is no teaching or suggestion that plug-in module N can be started until after completion of plug-in module M. The cited passage therefore does not recite signaling functionality between plug-in modules that depend on each other. Applicants respectfully submit that claim 34 includes limitations not taught or suggested by the cited prior art.

Claims 36-43

To expedite prosecution of the present application, Applicants submit new claims 36-43. Support for claims 36-42 can be found at page 6 line 6 to page 7 line 4 as well as page 17 line 18 to page 18 line 5 as well as elsewhere throughout the specification. Support for claim 43 can be found at FIG. 1 and corresponding text at pages 10-12. In view of the cited references, these claims include further limitations over the cited prior art.

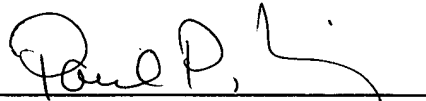
CONCLUSION

In view of the foregoing remarks, Applicants submit that the pending claims as well as newly added claims are in condition for allowance. A Notice to this effect is respectfully requested. If the Examiner believes, after reviewing this Response, that the pending claims are not in condition for allowance, the Examiner is respectfully requested to call the Representative.

-30-

Applicants hereby petition for any extension of time which is required to maintain the pendency of this case. If there is a fee occasioned by this response, including an extension fee, that is not covered by an enclosed check, please charge any deficiency to Deposit Account No. 50-3735.

Respectfully submitted,

A handwritten signature in black ink, appearing to read "Paul P. Kriz", is written over a horizontal line.

Paul P. Kriz, Esq.  
Attorney for Applicant(s)  
Registration No.: 45,752  
Chapin Intellectual Property Law, LLC  
Westborough Office Park  
1700 West Park Drive  
Westborough, Massachusetts 01581  
Telephone: (508) 616-9660  
Facsimile: (508) 616-9661

Attorney Docket No.: EMC01-11(01046)

Dated: July 11, 2006